대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 특허출원 2001년 제 81106 호
Application Number PATENT-2001-0081106

출 원 년 월 일 : 2001년 12월 19일
Date of Application DEC 19, 2001

출 원 인 : 한국전자통신연구원
Applicant(s) KOREA ELECTRONICS & TELECOMMUNICATIONS RESEARCH IN

2002 년 01 월 08 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0005
【제출일자】	2001.12.19
【발명의 명칭】	절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치
【발명의 영문명칭】	Method and apparatus for wrapping existing procedure oriented program into component based system
【출원인】	
【명칭】	한국전자통신연구원
【출원인코드】	3-1998-007763-8
【대리인】	
【성명】	권태복
【대리인코드】	9-2001-000347-1
【포괄위임등록번호】	2001-057650-1
【대리인】	
【성명】	이화익
【대리인코드】	9-1998-000417-9
【포괄위임등록번호】	1999-021997-1
【발명자】	
【성명의 국문표기】	이문수
【성명의 영문표기】	LEE,Moon-Soo
【주민등록번호】	730808-1822523
【우편번호】	305-345
【주소】	대전광역시 유성구 신성동 142-11번지 202호
【국적】	KR
【발명자】	
【성명의 국문표기】	김철홍
【성명의 영문표기】	KIM,Chul-Hong
【주민등록번호】	571006-1408410
【우편번호】	305-755

【주소】 대전광역시 유성구 어은동 99번지 한빛아파트 103동 105호

【국적】 KR

【발명자】

【성명의 국문표기】 양영종

【성명의 영문표기】 YANG,Young-Jong

【주민등록번호】 560308-1010433

【우편번호】 305-755

【주소】 대전광역시 유성구 어은동 한빛아파트 117동 902호

【국적】 KR

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인
권태복 (인) 대리인
이화익 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	19	면	19,000 원
【우선권주장료】	0	건	0 원
【심사청구료】	15	항	589,000 원

【합계】 637,000 원

【감면사유】 정부출연연구기관

【감면후 수수료】 318,500 원

【첨부서류】 1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 기존 절차 지향 프로그램을 컴포넌트 기반의 프로그램과 연계하기 위한 방법 및 그 장치에 관한 것으로, 기존 절차 중심의 언어로 구현된 프로그램을 파싱하여 재사용 가능한 비즈니스 로직을 식별하는 기능을 제공한다. 그리고 래핑 모듈은 식별되어진 모듈, 즉 비즈니스 로직을 프레임워크 기반의 소프트웨어 래핑 기법을 컴포넌트 기반의 프로그램과 연계하는 기능을 제공하는 것이다. 따라서, 본 발명은, 현재 운용되고 있는 절차 지향 프로그램을 소스 코드 변경 없이 새로운 컴포넌트 기반의 소프트웨어 개발 기법과 쉽게 연계할 수 있게 해 줌으로써 기존 프로그램의 재사용할 수 있게 도와주고 새로운 기능이 추가될 경우에도 용이하게 변경할 수 있는 것이다.

【대표도】

도 2

【색인어】

레거시, 컴포넌트, 래퍼, 워크플로우, 파라그래프, 제어변수,

【명세서】

【발명의 명칭】

절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치{Method and apparatus for wrapping existing procedure oriented program into component based system}

【도면의 간단한 설명】

도 1는 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치의 블록구성을 나타낸 도면.

도 2는 도 1에 도시된 비지니스 로직 식별기에서 수행되는 로직 식별 방법에 대한 동작 플로우챠트를 나타낸 도면.

도 3a는 도 2에서 비즈니스 로직을 식별하기 위한 구성 요소를 나타낸 도면.

도 3b는 도 3a에 도시된 로직 식별 구성 요소를 설정하기 위한 화면 구성을 예시한 도면.

도 4는 도 2에 도시된 프로그램 내부 흐름 서치의 구체적인 방법에 대한 동작 플로우챠트.

도 5a는 도 4의 알고리즘에 대한 소스 예를 나타낸 도면.

도 5b는 도 5a의 소스를 이용한 도 4의 호출 트리(Call Tree)의 화면 구성을 예시한 도면.

도 6는 도 2에 도시된 입출력 변수 서치 단계에서의 서치의 구체적인 방법에 대한 동작 플로우챠트.

도 7는 도 2에 도시된 비즈니스 로직의 워크플로우(workflow)를 식별하는 단계의 구체적인 동작 플로우챠트를 나타낸 도면.

도 8은 도 1의 컴포넌트 래퍼(Wrapper) 생성기의 상세 블록 구성을 나타낸 도면.

도 9는 도 8에 도시된 컴포넌트 래퍼의 중간 프레임워크 상세 블록 구성을 나타낸 도면.

도면의 주요부분에 대한 부호의 설명

110 : 코드 분석기 120 : 비지니스 로직 식별기

130 : 컴포넌트 래퍼 생성기 800 : 컴포넌트 프레임 워크

810 : 중간 프레임 워크 830 : 레거시 프레임 워크

900 : 프로그램 스케쥴러 910 : 레코드 핸들러

920 : 메타데이터 저장소 930 : 레코드 어댑터

950 : 레거시 컴포넌트

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<19> 본 발명은 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치에 관한 것으로서, 특히 식별된 모듈은 컴포넌트 래핑을 위하여 워크플로우(workflow)를 중심으로 식별하고 프레임워크 기반의 래퍼(Wrapper)를 이용하여 재사용 가능한 컴포넌트로 생성하기 위한 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치에 관한 것이다.

<20> 레거시 시스템(Lagacy System)은 기업 전략에 중요한 프로그램이고 수 십년 동안 COBOL이나 FORTRAN과 같은 절차적 언어로 구현되었다. 그리고 계속되는 수정과 기능 추가로 인해 문서화가 제대로 이루어지지 못하고 프로그램 인력 부족으로 인하여 유지 관리 비용이 계속해서 증가되고 있다.

<21> 또한, 기업용 소프트웨어는 시장의 요구 사항이 변화됨에 따라 계속해서 수용할 수 있어야 하지만 레거시 시스템은 큰 걸림돌이 되고 있다.

<22> 레거시 시스템을 새로운 소프트웨어 기법을 적용하기 위한 방법으로 재개발(Redevelopment) 방법, 변환(Transformation) 방법, 래핑(Wrapping)방법을 들 수 있다.

<23> 재개발 방법은, 기존 시스템과는 별개로 새로운 소프트웨어 기법을 이용하여 개발하게 된다. 상대적으로 시간, 비용과 안정성에 문제점을 가지고 있다.

<24> 변환 방법은, 기존 프로그램을 객체 지향이나 컴포넌트 기반의 소프트웨어 기법으로 재 추상화하고 이 정보를 이용하여 시스템을 새롭게 만들게 된다. 그리고 변환 방법은 재개발보다는 기존 시스템의 정보를 이용하므로 시간과 비용을 절감할 수 있지만 여전히 프로그램의 안정성에 대한 문제점을 여전히 내재하고 있다.

<25> 마지막 래핑 방법은, 앞의 예제와는 달리 기존 시스템은 계속 유지를 시키는 반면에 새로운 소프트웨어 기법을 용이하게 재사용할 수 있다는 장점을 가지고 있다. 하지만 소프트웨어 진화에 있어서 오히려 이러한 래핑 방법은 프로그램의 복잡도를 증가시키는 원인이 될 수도 있다.

<26> 래핑의 근본적인 문제점으로 과거의 래핑 기법은 단순히 사용자 인터페이스(User Interface)를 새로운 인터페이스로 현대화(Modernization)에 집중되어 단지 현재 시스템의 재사용하는데 초점이 맞추어진다.

<27> 따라서 래핑 결과물은 추후 재사용하려 할 때 오히려 복잡도를 증가시켜 주게 되었다. 이러한 문제점의 최선의 방법은 래핑의 대상이 단순히 사용자 인터페이스를 대상으로 하는 것이 아니라 실제로 추후에 재사용 가능성이 높은 부분을 식별하여 이를 래핑하는 것이다.

<28> 그리고 래퍼의 구조도 프레임워크를 기반으로 개발하여 유지 보수가 용이하도록 이루어져야 한다.

<29> 또한 래핑된 결과물은 하나의 기능을 가지는 독립된 컴포넌트로 만들어져야 한다. 왜냐하면 컴포넌트 개발 방법론(Component-Based Development, CBD)은 레거시

시스템을 그대로 포함할 수 있으므로 독립적인 기능을 가지고 있는 컴포넌트는 소프트웨어의 재사용성, 기능 확장뿐만 아니라 웹과 같은 분산 환경에 쉽게 이용될 수 있는 이점을 가지고 있다.

<30> 본 발명과 관련된 선행 기술로는 기존 시스템의 현대화(Modernization)에 관한 기술들과 유사한 분야인 소프트웨어 래핑 기술들을 들 수 있다. 소프트웨어 래핑에 관련된 기술로 기존 시스템의 데이터와 자료 구조를 객체 지향 방법론을 이용하여 객체 래핑한 경우(미국 특허 6305007), 메인 프레임에서 운용되고 있는 CICS COBOL프로그램을 ECI(External Call Interface) API를 이용하여 이종 운용체제 환경에서 접근할 수 있는 래퍼를 생성하는 경우(미국특허 6230117), 비즈니스 로직이 많이 포함되어 있는 기존 어플리케이션에 사용되는 화면들의 메타 정보를 모두 가지고 있으므로 해서 하나의 어플리케이션 전체를 위한 래퍼를 생성하는 특허(미국 특허 6253244)등이 있다.

<31> 이러한 선행 기술 대부분은 기존 프로그램을 새로운 시스템이나 프로그램 환경으로 연계시키기 위한 방법에 초점이 맞추어져 있으며 생성된 결과물의 소프트웨어 진화에 대한 고려는 포함하고 있지 않다.

<32> 래핑의 대상이 되는 기존 프로그램의 클러스터링이나 식별 과정은 사용자의 경험이나 직관과 같은 수작업이 필요하다.

<33> 따라서 기존 시스템을 새로운 프로그램 환경으로 이전할 경우에 재사용성이 높은 부분을 식별해 주는 기능과, 식별된 모듈을 래핑하는 과정이 연속적(Seamless)으로 이루어져야 한다. 그리고 래핑 결과물이 추후 유지 보수 및 소프트웨어 진

화의 용이성을 위해 일관된 아키텍처, 즉 프레임워크 기반의 래퍼 구조가 필요한 것이다.

【발명이 이루고자 하는 기술적 과제】

<34> 따라서, 본 발명은 상기한 종래 기술에 따른 문제점을 해결하기 위하여 안출한 것으로, 본 발명의 목적은, 개발자가 수작업으로 소스코드를 분석하고 경험과 직관을 이용하여 재사용 모듈을 식별하고, 이를 래핑하는 과정을 도구 지원을 통해 최대한 자동화 해주고, 보다 체계적인 방법을 제공하기 위한 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치를 제공함에 있다.

<35> 또한, 본 발명의 다른 목적은, 상기한 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법을 수행하기 위한 프로그램을 기록하여 컴퓨터로 읽을 수 있도록 한 기록매체를 제공함에 있다.

【발명의 구성 및 작용】

<36> 상기한 목적을 달성하기 위한 본 발명은 기존 시스템에서 재사용 가능한 기능(function)을 식별하는 식별 알고리즘에 있어서, 사용자가 그 시스템에 대한 세부 지식이 없을지라도 유스케이스(Use Case)와 같은 시스템의 전반적인 지식만으로 기본적인 구성요소의 가중치를 조절함으로써, 용이하게 비즈니스 로직을 하향식(Top-down)으로 식별하고, 식별된 비즈니스 로직을 컴포넌트 래핑을 위하여 상

향식(Bottom-up식으로 시스템의 워크플로우(workflow)를 자동 식별하여 필요한 제약 조건(constraint)과 외부 인터페이스(interface)를 자동 생성해 준다는 것을 특징으로 한다. 그리고 프레임워크 기반의 중간 프레임워크(Intermediate Framework)를 사용하여 보다 체계적이고 유지 보수가 용이하도록 컴포넌트 래핑 프로세스를 제공함에 그 특징이 있다.

<37> 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치의 일 측면에 따르면, 원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출해 내는 코드 분석부; 상기 코드 분석부에서 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 비지니스 로직 식별부; 상기 비지니스 로직 식별부에서 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 컴포넌트 래퍼 생성부를 포함한다.

<38> 상기 컴포넌트 래퍼 생성부는, 기존 시스템을 컴포넌트로 재사용하기 위한 컴포넌트 프레임 워크; 상기 컴포넌트 프레임워크와 연계될 시스템의 프레임워크인 레거시 프레임워크; 상기 컴포넌트 프레임워크와 상기 레거시 프레임워크를 서로 연결하고, 상기 레거시 프레임워크와 입출력이 일어나는 화면 정보를 캡쳐하여 정보를 자동으로 삽입하거나 추출하는 중간 프레임 워크를 포함한다.

<39> 그리고, 상기 중간 프레임 워크는, 프로그램과 프로그램간의 항해 정보와 상호 작용 관계를 가지고 있으면서 복수의 화면이 입력용인지 출력용인지에 대한 스케쥴 정보를 가지고 있는 프로그램 스케쥴러; 미리 등록된 하나의 워크플로우에 포함되어 있는 프로그램들의 화면에 대한 메타 정보를 저장하고 있는 메타데

이터 저장소; 상기 컴포넌트 프레임워크에서 요구하는 명령을 해석하여 상기 메타 데이터 저장소로부터 입출력 데이터의 메타 정보를 얻어내어 현재 기존 시스템으로부터 들어온 화면이 어떤 화면이고 그에 따른 입출력 데이터가 어떤 것이 있다는 것을 알아내어 입출력 데이터를 전달하는 레코드 핸들러; 레거시 컴포넌트로부터 들어오는 입력 화면들을 받아서 입출력에 관련된 데이터와 단순히 화면의 디스플레이를 위한 정보를 구별하여 상기 레코드 핸들러로 정보를 제공하는 레코드 어댑터를 포함하는 것이다.

<40> 한편, 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법의 일 측면에 따르면, 원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출하는 단계; 상기 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 단계; 상기 식별된 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 단계를 포함한다.

<41> 상기 식별하는 단계는, 식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 구성 요소들의 가중치 값을 입력받아 각 모듈의 규모에 따라 사용자에게 입력을 받은 구성 요소의 가중치 값을 이용하여 사용자의 요구 사항에 대한 적합 지수를 계산하는 단계; a) 상기 계산된 적합 지수가 최대 인지를 판단하여 적합 지수가 최대인 경우, 적합 지수가 최대인 모듈이 속해 있는 프로그램에서 이 모듈을 실행하기 위한 프로그램 내부의 흐름들을 서치하고, b) 사용자와 직접적으로 관계를 가지는 화면 장식에 관련된 변수를 중심으로 입출력 변수를 서치

하는 단계; 상기 서치된 프로그램 내부의 흐름(경로)들과 입출력 변수들을 이용하여 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계; 상기 식별된 변수들을 이용하여 제약조건으로 될 변수와 인터페이스가 될 변수를 정의하여 상기 래핑하기 위한 코드 생성을 위해 제공하는 단계를 포함할 수 있다.

<42> 상기 적합 지수를 계산하는 단계에서, 적합 지수의 계산은 규모가 큰 부분을 시작점으로 해서 작은 부분으로 검색해 나가는 하향식(Top-down)으로 진행해 가면서 사용자의 요구 사항에 대한 적합도(적합 지수)를 계산한다. 그리고, 상기 제약조건은 원하는 비즈니스 로직의 모듈을 실행하기 위한 흐름을 얻기 위해 필요한 제어 변수들로 구성되고 인터페이스는 데이터의 입출력 부분에 사용된 변수들로 구성된다.

<43> 상기 프로그램 내부의 흐름과 입출력 변수를 서치하는 단계에서, 프로그램 내부 흐름의 서치방법은, 프로그램 상세 분석을 위한 프로그램과 비즈니스 로직이 있는 최소한의 모듈, 파라그래프 후보를 이미 알고 있는 경우, 프로그램 내부 정보를 체계적으로 정리하기 위해 프로그램 내부를 모듈, 파라그래프 단위로 파라그래프간의 흐름 정보와 그에 따른 조건들에 관한 정보를 수집하는 단계; 각 모듈의 호출관계를 검색하기 위해서 함수 호출 명령문을 이용하여 각 파라그래프간의 호출 관계를 검색하는 단계; 상기 파라그래프 호출의 중복성이나 재귀적인 부분을 제거하고 호출의 포함 관계가 있을 경우에 이들 파라그래프 호출 관계를 재구성하는 단계; 비구조화 명령문에 의한 파라그래프 흐름을 검색하기 위해서 파라그래프간의 흐름 정보를 비구조화 명령문 문장들만 고려하여 프로그램의 흐름을 식별하는 단계; 상기 얻어진 파라그래프의 호출 관계 정보와 상기 비구조화

문장에 의한 프로그램 흐름 정보를 이용하여 호출관계 트리를 생성하는 단계를 포함한다.

<44> 상기 프로그램 내부의 흐름과 입출력 변수를 서치하는 단계에서, 입출력 변수를 서치하는 방법은, 이미 재사용하고자 하는 비즈니스 로직을 포함하고 있는 프로그램 내부에 내재되어 있는 입출력 변수나 사용자 인터페이스에 대한 정보 또는 화면을 표시하기 위한 폼을 가지고 있는 파일을 분석하여 각 변수, 필드에 대한 화면 정보를 분석하는 단계; 상기 분석된 화면 정보에서 필드가 존재하는지를 판단하는 단계; 상기 판단 결과, 분석된 화면 정보에서 필드가 존재하는 경우에는 필드가 실제 데이터를 입출력하기 위한 부분인지 아니면 단순한 화면을 장식하기 위한 부분인지를 판별하는 단계; a) 상기 필드가 입출력(I/O) 필드에 해당되는 경우, 필드가 입출력 변수로 이용되기 때문에 입출력 변수로 등록하고, b) 필드가 입출력을 위한 부분에 해당하지 않는 경우에는 필드가 화면을 장식하는 용도로 사용되기 때문에 메타 데이터로 등록하는 단계를 포함할 수 있다.

<45> 상기 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계는, 생성된 트리에서 사용자가 원하는 워크 플로우를 가지고 있는 유일한 경로(Unique Path)를 선택하는 단계; 지정된 워크플로우 중에서 파라그래프의 흐름이나 입출력을 결정하는 변수 등 중요한 변수(Critical variable)가 있는지 여부를 검사하는 단계; 검사 결과, 중요한 변수가 존재하는 경우, 영향 분석를 이용하여 중요한 변수에 영향을 주게 된 변수들의 리스트를 추적하거나 프로그램 간에 전달되는 변수일 경우에는 호출하거나 호출을 당한 프로그램으로 계속해서 추적하여 변수의 용도를 식별해 내는 단계; 상기 식별된 변수가 워크플로우의 경로를

결정짓는 제어변수인지 제약조건에 이용되는 변수인지를 판별하는 단계; a) 상기 판단 결과, 상기 식별된 변수가 제어 변수에 이용되는 경우, 제어변수 리스트에 추가 시키고, b) 반대로 만약 식별된 변수가 제약조건으로 사용된 경우 제약조건 리스트에 추가시키는 단계를 포함한다.

<46> 한편, 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법을 수행하기 위하여 디지털 처리장치에 의해 실행될 수 있는 명령어들의 프로그램이 유형적으로 구현되어 있으며, 디지털 처리장치에 의해 판독될 수 있는 기록 매체의 일 측면에 따르면, 원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출하는 단계; 상기 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 단계; 상기 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 단계를 포함하되, 상기 식별하는 단계는, 식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 구성 요소들의 가중치 값을 입력받아 각 모듈의 규모에 따라 사용자에게 입력을 받은 구성 요소의 가중치 값을 이용하여 사용자의 요구 사항에 대한 적합 지수를 계산하는 단계; a) 상기 계산된 적합 지수가 최대 인지를 판단하여 적합 지수가 최대인 경우, 적합 지수가 최대인 모듈이 속해 있는 프로그램에서 이 모듈을 실행하기 위한 프로그램 내부의 흐름들을 검색하여 프로그램 내부의 흐름들을 서치하고, b) 사용자와 직접적으로 관계를 가지는 화면 장식에 관련된 변수를 중심으로 입출력 변수를 서치하는 단계; 상기 서치된 프로그램 내부의 흐름(경로)들과 입출력 변수들을 이용하여 제

약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계; 상기 식별된 변수들을 이용하여 제약조건으로 될 변수와 인터페이스가 될 변수를 정의하여 상기 래핑하기 위한 코드 생성을 위해 제공하는 단계를 수행한다.

<47> 이하, 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치에 대한 바람직한 일 실시예를 첨부한 도면을 참조하여 상세하게 살펴보기로 하자,

<48> 도 1은 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치에 대한 블록 구성을 나타낸 도면으로서, 코드 분석기(110), 비지니스 로직 식별기(120) 및 컴포넌트 래퍼 생성기(130)로 구성될 수 있다.

<49> 코드 분석기(110)는 원시 절차적 언어로 구현된 소스 프로그램 또는 코드(100)들에서 프로그램 분석에 필요한 정보를 추출해 내어 비지니스 로직 식별기(120)로 제공한다.

<50> 비지니스 로직 식별기(120)는, 상기 코드 분석기(110)에서 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별한다.

<51> 컴포넌트 래퍼 생성기(130)는 상기 비지니스 로직 식별기(120)에서 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우(workflow)를 래핑하기 위한 코드를 자동 생성하는 것이다.

<52> 상기한 비지니스 식별기(120)에 대한 구체적인 동작에 대하여 도 2를 참조하여 상세하게 설명해 보기로 하자. 도 2는 도 1에 도시된 비지니스 로직 식별기에서 수행되는 로직 식별 방법에 대한 동작 플로우챠트를 나타낸 도면이다.

<53> 먼저, 식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 구성요소들의 가중치 값을 입력받게 된다(S200).

<54> 각 모듈의 규모에 따라 사용자에게 입력을 받은 구성 요소의 가중치 값을 이용하여 식별한다(S200). 즉, 모듈의 규모는 프로그램, 파라그래프(paragraph)나 함수 단위로 각 절차적 언어에서 제공되는 물리적으로 독립되어 분리 될 수 있거나 최소한으로 모듈화가 가능한 것들이 될 수 있다.

<55> 규모가 큰 부분을 시작점으로 해서 작은 부분으로 검색해 나가는 하향식(Top-down)방식으로 진행해 가면서 사용자의 요구 사항에 대한 적합도(적합 지수)를 계산하게 된다(S210).

<56> 이렇게 계산된 적합 지수가 최대인지를 판단하고(S220), 판단 결과, 적합 지수가 최대인 경우, 적합 지수가 최대인 모듈이 속해 있는 프로그램에서 이 모듈을 실행하기 위한 프로그램 내부의 흐름들을 검색하여 가능한 모든 경우의 경로들, 즉 프로그램 내부의 흐름들을 찾아 낸다(S230).

<57> 그러나, 상기 S220단계에서, 계산된 적합 지수가 최대가 아닌 경우에는 사용자가 직접 선택할 수도 있다.

<58> 이어, 상기 사용자와 직접적으로 입출력 관계를 가지는 화면 장식에 관련된 변수를 중심으로 입출력 변수를 찾아 낸다(S240).

<59> 이어, 상기 찾아낸 프로그램 내부의 경로들과 입출력 변수들을 이용하여 하나의 프로그램 내부의 흐름을 식별해내고 여기에 필요한 제약조건(constraint)과 인터페이스에 관련된 변수를 찾아 낸다. 제약조건은 원하는 비즈니스 로직의 모듈을 실행하기 위한 흐름을 얻기 위해 필요한 제어 변수들로 구성되고 인터페이스는 데이터의 입출력 부분에 사용된 변수들로 구성된다.

<60> 따라서 상향식(Bottom-up)으로 프로그램 간의 워크플로우(workflow)를 식별할 때 이 부분에서도 비즈니스 로직을 포함하고 있는 프로그램을 실행할 수 있기 위한 변수들과 화면 입출력에 관련된 변수들을 이용하여 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 낸다(S250).

<61> 이어, 상기 S250 단계의 워크플로우 식별에서 생성된 변수들을 이용하여 제약조건으로 될 변수와 인터페이스가 될 변수를 정의하게 된다(S260).

<62> 따라서, 도 1에 도시된 컴포넌트 래퍼 생성기(130)는 최종적으로 상기 식별된 워크플로우(workflow)를 이용하여 프레임워크 기반의 래퍼를 생성하기 위한 코드를 자동 생성해 주게 되는 것이다(S270).

<63> 도 3a는 도 2의 S200단계에서, 식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 입력되는 구성 요소의 리스트 테이블이다.

<64> 도 3a에 도시된 바와 같이, 프로그램의 용도(Program)(300)는 프로그램 수준에 있어서 프로그램의 용도에 관한 것으로 외부 프로그램 호출(External Call)(301), 화면의 입출력(Screen I/O)(302)으로 구성된다.

<65> 수학적 용도(Mathematics)(310)는 하나의 문장(Statement)에서 입출력 변수들의 연산, 즉 값을 할당하거나 계산에 관련된 것이다. 이 수학적 용도(310)는 입력에 관련된 변수가 다른 변수에 값을 할당(Math Input)(311), 출력에 관련된 변수로 값을 치환(Math Output)(312)과 입출력 변수의 계산(Math Operation)(313)으로 구성된다.

<66> 데이터 집합의 용도(Data Set)(320)는 데이터를 파일이나 다른 데이터베이스에 저장하는 것에 관련된 것이다. 데이터 집합의 용도(320)는 읽기(Read)(311),쓰기(Write)(322), 삭제(Delete)(323), 갱신(Update)(324) 등으로 구성된다.

<67> 도 3b는 도 3a의 각 구성 요소를 설정하기 위한 화면 구성을 예시한 것으로, 비즈니스 로직의 구성요소에 대한 설정 화면(340)이다.

<68> 사용자는 원하는 각 요소의 가중치를 조절함으로써, 자동으로 사용자가 원하는 비즈니스 로직을 식별해 주게 된다. 예를 들어 설명하면, 만약 사용자가 기존 시스템에서 조회 기능에 관련된 비즈니스 로직을 찾으려고 한다면 사용자는 화면 입출력(302), 출력 변수로 값을 치환(312), DataSet 읽기(311)의 가중치를 높게 설정하면 되는 것이다.

<69> 도 4는 도 2에 도시된 프로그램 내부 흐름 서치의 구체적인 방법에 대한 동작 플로우챠트이다.

<70> 도 2의 S250 단계인 프로그램 내부 흐름 검색을 실행하는 단계에서는 이미 재 사용하고자 하는 비즈니스 로직을 포함하고 있는 프로그램을 알고 있다.

<71> 따라서 도 2에 도시된 S250 단계의 프로그램 내부 흐름 검색을 위한 사전 조건들로써, 상세 분석을 위한 프로그램과 비즈니스 로직이 있는 최소한의 모듈, 파라그래프 후보를 이미 알고 있다고 가정한다(S410).

<72> 이어, 프로그램 내부 정보를 체계적으로 정리하기 위해 프로그램 내부를 모듈, 파라그래프 단위로 파라그래프간의 흐름 정보와 그에 따른 조건(Condition)들에 관한 정보를 수집한다(S410).

<73> 이어, 각 모듈의 호출관계를 검색하기 위해서 COBOL에서는 CALL문과 PERFORM문과 같은 함수 호출 명령문을 이용하여 각 파라그래프간의 호출 관계를 검색하게 된다(S420).

<74> 그리고, 호출 관계를 정제(Refinement)하기 위해서 S430에서는 파라그래프 호출의 중복성이나 재귀적인(Recursive) 부분을 제거하고 호출의 포함 관계가 있을 경우에 이들 포함관계를 재구성해 주게 된다(S430).

<75> 이어, 비구조화 명령문에 의한 파라그래프 흐름을 검색하기 위해서 파라그래프간의 흐름 정보를 GO TO문, CONTINUE문, BREAK문 등과 같은 비구조(Unstructure)화 명령문 문장들만 고려하여 프로그램의 흐름을 식별한다(S440).

<76> 이어, 상기 생성된 각 파라그래프의 호출관계를 표현하는 호출 트리(Call Tree)를 생성한다(S450). 즉, S430 단계에서 얻어진 파라그래프의 호출 관계 정보와 S440 단계의 비구조화 문장에 의한 프로그램 흐름 정보를 모두 이용하여 호출관계 트리를 생성해 주는 것이다.

<77> 도 5a는 도 4의 알고리즘의 소스 예를 나타낸 도면이고, 도 5b는 도 5a의 소스를 이용한 도 4의 호출 트리(Call Tree)의 화면 구성을 예시한 도면이다.

<78> 즉, 도 5b는 도 4의 알고리즘을 이용하여 도 5a에 도시된 프로그램 소스를 분석하고 호출 관계 트리(Call Tree)를 화면으로 표시한 것이다.

<79> 호출 관계 트리의 메타 데이터는 XML형식으로 표현되어 있다. <NAME>태그는 파라그래프 이름(520)을 나타내고, <CTL>태그는 다른 파라그래프를 호출하거나 항해하기 위한 제어조건(Control Condition) 리스트(540)를 나타낸다. 또한,
태그는 호출되거나 다음에 항해할 파라그래프 리스트(550)를 나타낸다.

<80> 호출되거나 다음에 항해할 파라그래프 리스트(550)는 이름 첫 문자가 '%'로 시작하는 파라그래프 이름(530)은 GOTO문과 같은 비구조화 문장을 표시한 것이다

.

<81> 도 6은 도 2에 도시된 입출력 변수 서치 단계에서의 서치의 구체적인 방법에 대한 동작 플로우챠트이다. 즉, 도 6은 도 2의 프로그램의 외부, 즉 화면정보와 관련된 입출력 변수를 검색하는 방법의 순서도이다.

<82> 도 6에 도시된 바와 같이, 이미 재사용하고자 하는 비즈니스 로직을 포함하고 있는 프로그램을 알고 있기 때문에 이 프로그램과 관련된 화면 정보나 파일을 알 수 있다(S600).

<83> 프로그램 내부에 내재되어 있는 입출력 변수나 사용자 인터페이스에 대한 정보 또는 화면을 표시하기 위한 폼(Form)을 가지고 있는 파일을 분석하여 각 변수, 필드에 대한 화면 정보를 분석한다(S610).

<84> 이어, 상기 분석된 화면 정보에서 필드가 존재하는지를 판단하고(S620), 판단 결과, 상기 분석된 화면 정보에서 필드가 존재하는 경우에는 필드가 실제 데이터를 입출력하기 위한 부분인지 아니면 단순한 화면을 장식하기 위한 부분인지를 판별하게 된다(S630). 그리고, 상기 필드가 입출력(I/O) 필드에 해당되는지 판단하게 되는 것이다(S640).

<85> 상기 판단 결과, 필드가 입출력을 위한 부분에 해당되면, 필드가 입출력 변수로 이용되기 때문에 입출력 변수로 등록하고(S650), 반대로 필드가 입출력을 위한 부분에 해당하지 않는 경우에는 필드가 화면을 장식하는 용도로 사용되기 때문에 메타 데이터로 등록을 하게 된다(S660).

<86> 도 7은 도 2에 도시된 비즈니스 로직의 워크플로우(workflow)를 식별하는 단계의 구체적인 동작 플로우챠트를 나타낸 도면이다.

<87> 먼저, 도 4의 S450단계에서 생성된 트리에서 사용자가 원하는 워크 플로우를 가지고 있는 유일한 경로(Unique Path)를 선택하게 된다(S700).

<88> 이어, 지정된 워크플로우(workflow)중에서 파라그래프의 흐름이나 입출력을 결정하는 변수 등 중요한 변수(Critical variable)가 있는지 여부를 검사한다(S710).

<89> 검사 결과, 중요한 변수가 존재하는 경우, 영향 분석(Impact Analysis)을 이용하여 중요한 변수에 영향을 주게 된 변수들의 리스트를 추적하거나 프로그램간의 전달되는 변수일 경우에는 호출하거나 호출을 당한 프로그램으로 계속해서 추적하여 변수의 용도를 식별해 낸다(S720).

<90> 이어, 상기 식별된 변수가 워크플로우의 경로를 결정짓는 제어변수(Control Variable)인지 제약조건(constraint)에 이용되는 변수인지를 판별한다(S730).

<91> 판단 결과, 상기 식별된 변수가 제어 변수에 이용되는 경우, 제어변수 리스트에 추가시키고(S740), 반대로 만약 식별된 변수가 제약조건(constraint)으로 사용된 경우 제약 조건 리스트에 추가 시키고 이 리스트는 앞으로 생성될 컴포넌트의 인터페이스 후보가 된다(S750).

<92> 도 8은 도 1의 컴포넌트 래퍼(Wrapper) 생성기의 상세 블록 구성을 나타낸 도면이다.

<93> 컴포넌트 래퍼는 기존 시스템에서 추출된 재사용하고자 하는 워크플로우(workflow)를 컴포넌트로 래핑하기 위한 것으로 컴포넌트 래퍼가 포함되어진 컴포넌트 시스템의 전체 구성은 크게 컴포넌트 프레임워크(800), 중간 프레임워크(820), 레거시 프레임워크(830)로 나누어진다.

<94> 컴포넌트 프레임워크(800)는 기존 시스템을 컴포넌트로 재사용하기 위한 프레임워크이고, 레거시 프레임워크(830)는 컴포넌트 프레임워크(800)와 연계될 시스템의 프레임워크이다.

<95> 그리고, 중간 프레임워크(810)는 컴포넌트 프레임워크(800)와 레거시 프레임워크(830)를 서로 연결하기 위한 프레임워크로 레거시 프레임워크(830)와 입출력이 일어나는 화면 정보(820)를 캡쳐하여 정보를 자동으로 삽입하거나 추출해주는 역할을 한다.

<96> 도 9는 도 8에 도시된 컴포넌트 레퍼의 중간 프레임워크 상세 블록 구성을 나타낸 도면이다.

<97> 중간 프레임워크(810)는, 도 9에 도시된 바와 같이 프로그램 스케쥴러(900), 레코드 핸들러(910), 메타데이터 저장소(920), 레코드 어뎁터(930)로 구성된다.

<98> 프로그램 스케쥴러(900)는 프로그램과 프로그램간의 항해 정보와 상호작용(Interaction) 관계를 가지고 있다. 프로그램의 상호 작용 관계는 프로그램이 입력, 출력, 입출력 용도 중에서 어떻게 사용할 것인지에 대한 정보를 가지고 있다. 예를 들면 만약 기존 프로그램의 화면 구성이 첫번째는 메뉴판으로 구성되어 있고, 사용자가 조회 메뉴를 선택하면, 다음 화면은 조회한 내용을 보여주는 것으로 구성되어 있다고 한다면, 프로그램 스케쥴러(900)는 이러한 두 개의 화면 흐름 정보를 가지고 있을 뿐만 아니라 각 화면이 입력용인지 출력용인지에 대한 정보도 가지고 있다. 즉 메뉴 화면은 입력용으로 사용될 것이고 조회 결과 화면은 출력용으로 사용되는 것이다.

<99> 메타데이터 저장소(920)는 도 6의 S660 단계에서 등록된 하나의 워크플로우(workflow)에 포함되어 있는 프로그램들의 화면에 대한 메타 정보를 가지고 있다.

<100> 레코드 핸들러(910)는 도 8에 도시된 컴포넌트 프레임워크(800)에서 요구하는 명령을 해석하여 메타 데이터 저장소(920)로부터 입출력 데이터의 메타 정보를 얻어내어 현재 기존 시스템으로부터 들어온 화면이 어떤 화면이고 그에 따른

입출력 데이터가 어떤 것이 있다는 것을 알아내어 입출력 데이터를 전달하는 기능을 한다.

<101> 레코드 어뎁터(930)는 기존 시스템으로부터 들어오는 입력 화면들을 받아서 입출력에 관련된 데이터와 단순히 화면의 디스플레이를 위한 정보를 구별하여 레코드 핸들러(910)로 정보를 제공한다.

<102> 그리고, 레코드 어뎁터(930)는 기존 시스템의 정보를 임시 저장하는 역할뿐만 아니라 ASCII나 EBCDIC과 같은 서로 다른 문자를 변환해주는 기능을 제공한다.

【발명의 효과】

<103> 상기한 바와 같은 본 발명에 따른 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법 및 그 장치는, 기존 시스템 구현에 대한 상세한 지식이 없는 개발자도 일반적인 프로그램 구성 정보만을 이용하여 시스템의 중요한 비즈니스 로직을 자동 식별할 수 있게 함으로써 기존 시스템 이해에 걸리는 시간과 비용을 절감할 수 있다.

<104> 또한, 소프트웨어 래핑 기법을 이용함으로써 기존 시스템의 장점인 안정성을 계속 유지해 나갈 수 있을 뿐만 아니라 재사용 가능한 부분을 식별하여 컴포넌트 래핑을 함으로써 기존 시스템을 웹과 같은 새로운 환경의 시스템에서도 연계하여 재 사용하고 추가 요구사항이 생길 경우에는 기능의 추가가 용이하다.

<105> 그리고 제안된 소프트웨어 래핑 방법은 단순히 화면 부분을 래핑하는 스크린 스크래핑(Screen Scraping)이라기 보다는 프레임워크 기반의 래퍼 구조를 가지므로 해서 유지 보수 및 추후 소프트웨어 진화에 있어서 확장성이 용이해지는 효과를 가진다.

【특허청구범위】

【청구항 1】

절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치에 있어서,

원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출해 내는 코드 분석부;

상기 코드 분석부에서 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 비지니스 로직 식별부;

상기 비지니스 로직 식별부에서 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 컴포넌트 래퍼 생성부를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치.

【청구항 2】

제1항에 있어서,

상기 컴포넌트 래퍼 생성부는,

기존 시스템을 컴포넌트로 재사용하기 위한 컴포넌트 프레임 워크;

상기 컴포넌트 프레임워크와 연계될 시스템의 프레임워크인 레거시 프레임 워크;

상기 컴포넌트 프레임워크와 상기 레거시 프레임워크를 서로 연결하고, 상기 레거시 프레임워크와 입출력이 일어나는 화면 정보를 캡쳐하여 정보를 자동으로 삽입하거나 추출하는 중간 프레임 워크를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치.

【청구항 3】

제2항에 있어서,

상기 중간 프레임 워크는,

프로그램과 프로그램간의 항해 정보와 상호 작용 관계를 가지고 있으면서 복수의 화면이 입력용인지 출력용인지에 대한 스케쥴 정보를 가지고 있는 프로그램 스케쥴러;

미리 등록된 하나의 워크플로우에 포함되어 있는 프로그램들의 화면에 대한 메타 정보를 저장하고 있는 메타데이터 저장소;

상기 컴포넌트 프레임워크에서 요구하는 명령을 해석하여 상기 메타 데이터 저장소로부터 입출력 데이터의 메타 정보를 얻어내어 현재 기존 시스템으로부터 들어온 화면이 어떤 화면이고 그에 따른 입출력 데이터가 어떤 것이 있다는 것을 알아내어 입출력 데이터를 전달하는 레코드 핸들러;

레거시 컴포넌트로부터 들어오는 입력 화면들을 받아서 입출력에 관련된 데이터와 단순히 화면의 디스플레이를 위한 정보를 구별하여 상기 레코드 핸들러로

정보를 제공하는 레코드 어댑터를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치.

【청구항 4】

제3항에 있어서,

상기 레코드 어뎁터는,

레거시 컴포넌트의 정보를 임시 저장하고, ASCII나 EBCDIC과 같은 서로 다른 문자를 변환해 주는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치.

【청구항 5】

제3항에 있어서,

상기 프로그램 스케쥴러는, 프로그램이 입력, 출력, 입출력 용도 중에서 어떻게 사용할 것인지에 대한 정보를 저장하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 장치.

【청구항 6】

절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법에 있어서,

원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출하는 단계;

상기 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 단계;

상기 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 단계를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 7】

제6항에 있어서,

상기 식별하는 단계는,

식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 구성 요소들의 가중치 값을 입력받아 각 모듈의 규모에 따라 사용자에게 입력을 받은 구성 요소의 가중치 값을 이용하여 사용자의 요구 사항에 대한 적합 지수를 계산하는 단계;

a) 상기 계산된 적합 지수가 최대 인지를 판단하여 적합 지수가 최대인 경우, 적합 지수가 최대인 모듈이 속해 있는 프로그램에서 이 모듈을 실행하기 위한 프로그램 내부의 흐름들을 검색하여 프로그램 내부의 흐름들을 서치하고,

b) 사용자와 직접적으로 관계를 가지는 화면 장식에 관련된 변수를 중심으로 입출력 변수를 서치하는 단계;

상기 서치된 프로그램 내부의 흐름(경로)들과 입출력 변수들을 이용하여 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계;

상기 식별된 변수들을 이용하여 제약조건으로 될 변수와 인터페이스가 될 변수를 정의하여 상기 래핑하기 위한 코드 생성을 위해 제공하는 단계를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 8】

제7항에 있어서,

상기 적합 지수를 계산하는 단계에서, 적합 지수의 계산은

규모가 큰 부분을 시작점으로 해서 작은 부분으로 검색해 나가는 하향식(Top-down)방식으로 진행해 가면서 사용자의 요구 사항에 대한 적합도(적합 지수)를 계산하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 9】

제7항에 있어서,

상기 제약조건은 원하는 비즈니스 로직의 모듈을 실행하기 위한 흐름을 얻기 위해 필요한 제어 변수들로 구성되고 인터페이스는 데이터의 입출력 부분에

사용된 변수들로 구성되는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 10】

제7항에 있어서,

상기 프로그램 내부의 흐름과 입출력 변수를 서치하는 단계에서, 프로그램 내부의 흐름의 서치방법은,

프로그램 상세 분석을 위한 프로그램과 비즈니스 로직이 있는 최소한의 모듈, 파라그래프 후보를 이미 알고 있는 경우, 프로그램 내부 정보를 체계적으로 정리하기 위해 프로그램 내부를 모듈, 파라그래프 단위로 파라그래프간의 흐름 정보와 그에 따른 조건들에 관한 정보를 수집하는 단계;

각 모듈의 호출관계를 검색하기 위해서 함수 호출 명령문을 이용하여 각 파라그래프간의 호출 관계를 검색하는 단계;

상기 파라그래프 호출의 중복성이나 재귀적인 부분을 제거하고 호출의 포함 관계가 있을 경우에 이들 파라그래프 호출 관계를 재구성하는 단계;

비구조화 명령문에 의한 파라그래프 흐름을 검색하기 위해서 파라그래프간의 흐름 정보를 비구조화 명령문 문장들만 고려하여 프로그램의 흐름을 식별하는 단계;

상기 얻어진 파라그래프의 호출 관계 정보와 상기 비구조화 문장에 의한 프로그램 흐름 정보를 이용하여 호출관계 트리를 생성하는 단계를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 11】

제10항에 있어서,

상기 파라그래프간의 호출 관계를 검색하는 단계에서 함수 호출 명령문은, COBOL에서는 CALL문과 PERFORM문을 이용하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 12】

제10항에 있어서,

상기 프로그램의 흐름을 식별하는 단계에서, 비구조화 명령문은, GO TO문, CONTINUE문, BREAK문 중 적어도 하나의 문장을 이용한 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 13】

제7항에 있어서,

상기 프로그램 내부의 흐름과 입출력 변수를 서치하는 단계에서, 입출력 변수를 서치하는 방법은,

이미 재사용하고자 하는 비즈니스 로직을 포함하고 있는 프로그램 내부에 내재되어 있는 입출력 변수나 사용자 인터페이스에 대한 정보 또는 화면을 표시하기 위한 폼을 가지고 있는 파일을 분석하여 각 변수, 필드에 대한 화면 정보를 분석하는 단계;

상기 분석된 화면 정보에서 필드가 존재하는지를 판단하는 단계;

상기 판단 결과, 분석된 화면 정보에서 필드가 존재하는 경우에는 필드가 실제 데이터를 입출력하기 위한 부분인지 아니면 단순한 화면을 장식하기 위한 부분인지를 판별하는 단계;

a) 상기 필드가 입출력(I/O) 필드에 해당되는 경우, 필드가 입출력 변수로 이용되기 때문에 입출력 변수로 등록하고,

b) 필드가 입출력을 위한 부분에 해당하지 않는 경우에는 필드가 화면을 장식하는 용도로 사용되기 때문에 메타 데이터로 등록하는 단계를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 14】

제7항에 있어서,

상기 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계는,

생성된 트리에서 사용자가 원하는 워크 플로우를 가지고 있는 유일한 경로(Unique Path)를 선택하는 단계;

지정된 워크플로우중에서 파라그래프의 흐름이나 입출력을 결정하는 변수 등 중요한 변수(Critical variable)가 있는지 여부를 검사하는 단계;

검사 결과, 중요한 변수가 존재하는 경우, 영향 분석를 이용하여 중요한 변수에 영향을 주게 된 변수들의 리스트를 추적하거나 프로그램 간의 전달되는 변수일 경우에는 호출하거나 호출을 당한 프로그램으로 계속해서 추적하여 변수의 용도를 식별해 내는 단계;

상기 식별된 변수가 워크플로우의 경로를 결정짓는 제어변수인지 제약조건에 이용되는 변수인지를 판별하는 단계;

a) 상기 판단 결과, 상기 식별된 변수가 제어 변수에 이용되는 경우, 제어 변수 리스트에 추가 시키고,

b) 반대로 만약 식별된 변수가 제약조건으로 사용된 경우 제약 조건 리스트에 추가시키는 단계를 포함하는 절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법.

【청구항 15】

절차 지향 프로그램을 컴포넌트 기반의 시스템으로 래핑하기 위한 방법을 수행하기 위하여 디지털 처리장치에 의해 실행될 수 있는 명령어들의 프로그램이 유형적으로 구현되어 있으며, 디지털 처리장치에 의해 판독될 수 있는 기록 매체에 있어서,

원시 절차적 언어로 구현된 소스 프로그램 또는 코드들에서 프로그램 분석에 필요한 정보를 추출하는 단계;

상기 추출된 프로그램 분석에 필요한 정보를 이용하여 재사용 가능성이 높은 부분을 식별하는 단계;

상기 식별한 비즈니스 로직을 포함하고 있는 프로그램 워크플로우를 래핑하기 위한 코드를 자동 생성하는 단계를 포함하되,

상기 식별하는 단계는,

식별하고자 하는 비즈니스 유형을 표현하기 위해 사용자로부터 구성 요소들의 가중치 값을 입력받아 각 모듈의 규모에 따라 사용자에게 입력을 받은 구성 요소의 가중치 값을 이용하여 사용자의 요구 사항에 대한 적합 지수를 계산하는 단계;

a) 상기 계산된 적합 지수가 최대 인지를 판단하여 적합 지수가 최대인 경우, 적합 지수가 최대인 모듈이 속해 있는 프로그램에서 이 모듈을 실행하기 위한 프로그램 내부의 흐름들을 검색하여 프로그램 내부의 흐름들을 서치하고,

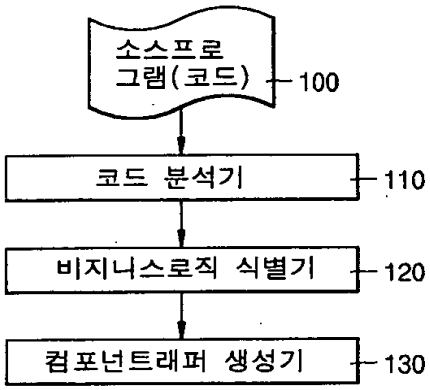
b) 사용자와 직접적으로 관계를 가지는 화면 장식에 관련된 변수를 중심으로 입출력 변수를 서치하는 단계;

상기 서치된 프로그램 내부의 흐름(경로)들과 입출력 변수들을 이용하여 제약 조건과 인터페이스에 필요한 변수를 자동 식별해 내는 단계;

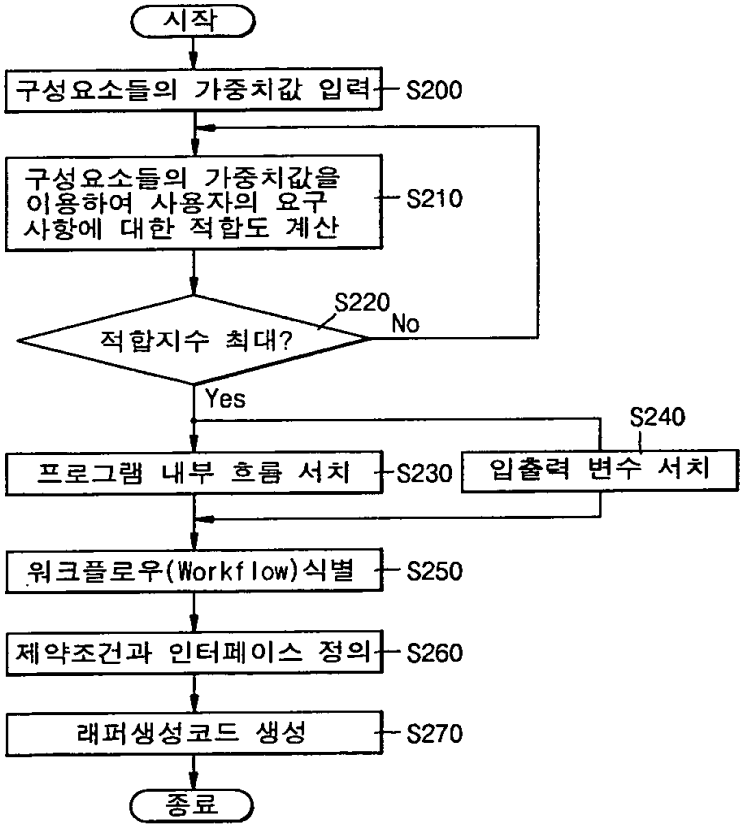
상기 식별된 변수들을 이용하여 제약조건으로 될 변수와 인터페이스가 될 변수를 정의하여 상기 래핑하기 위한 코드 생성을 위해 제공하는 단계를 수행하는 기록 매체.

【도면】

【도 1】


소스프로
그램(코드)
100
코드 분석기
110
비지니스로직 식별기
120
컴포넌트래퍼 생성기
130

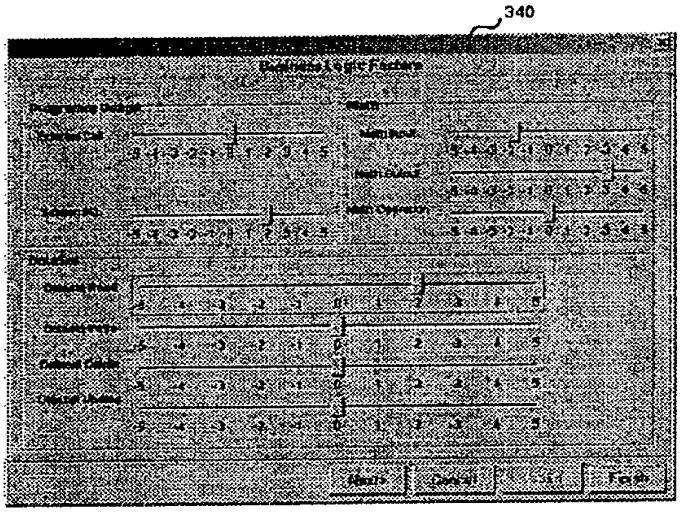
【도 2】


시작
구성요소들의 가중치값 입력
S200
구성요소들의 가중치값을
이용하여 사용자의 요구
사항에 대한 적합도 계산
S210
적합지수 최대?
S220
No
Yes
S240
프로그램 내부 흐름 서치
S230
입출력 변수 서치
워크플로우(Workflow)식별
S250
제약조건과 인터페이스 정의
S260
래퍼생성코드 생성
S270
종료

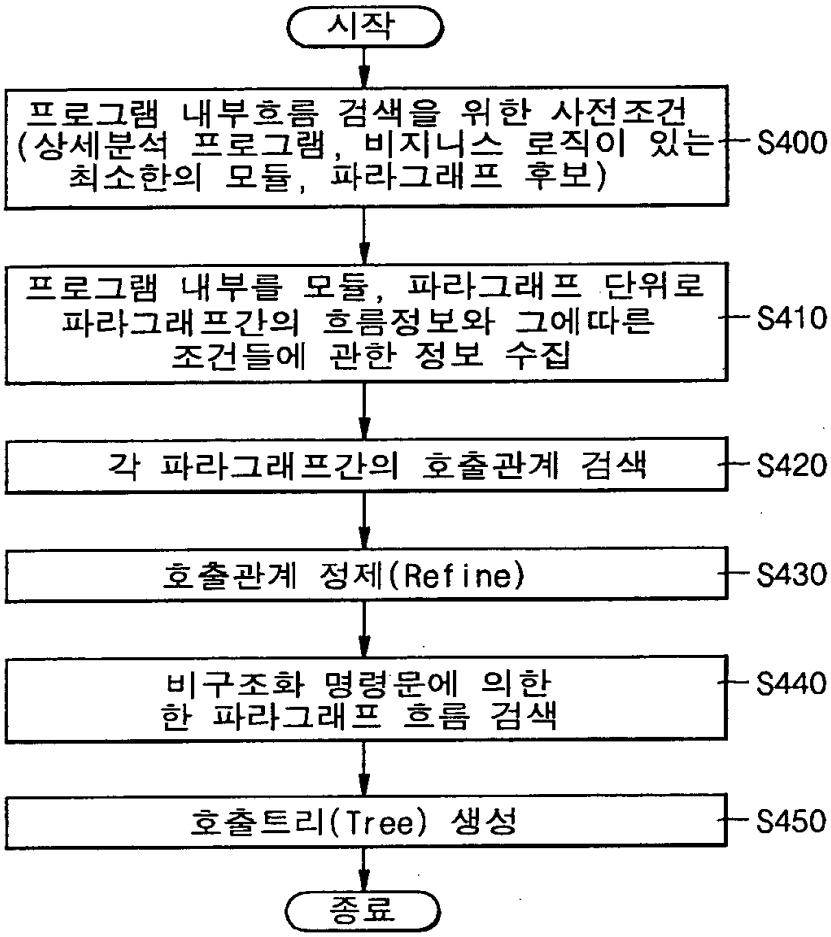
【도 3a】

Usage Group	Structural Elements	Weight
Program (300)	External Call	W1 (301)
	Screen I/O	W2 (302)
Mathematics (310)	Math Input	W3 (311)
	Math Output	W4 (312)
	Math Operation	W5 (313)
DataSet (320)	Read	W6 (321)
	Write	W7 (322)
	Delete	W8 (323)
	Update	W9 (324)

【도 3b】



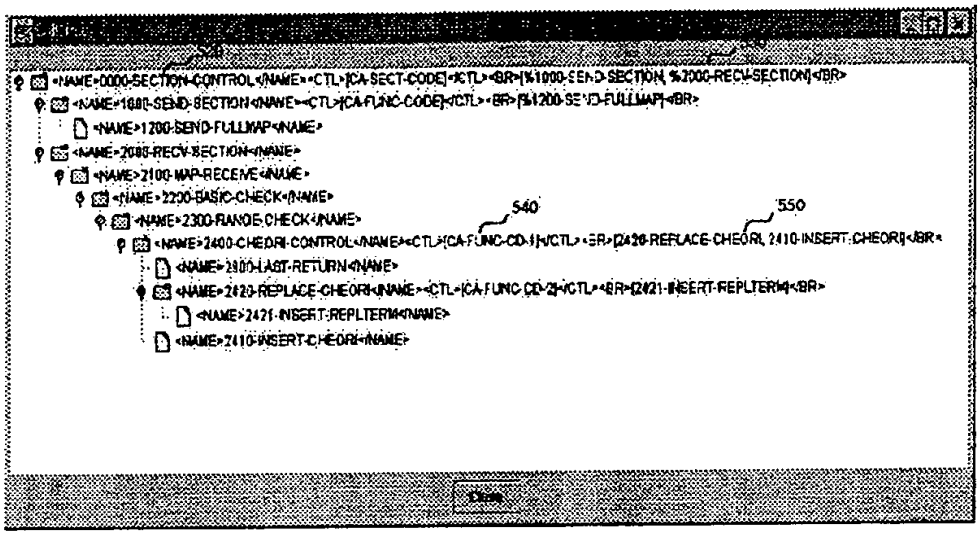
【도 4】


시작
프로그램 내부흐름 검색을 위한 사전조건 (상세분석 프로그램, 비지니스 로직이 있는 최소한의 모듈, 파라그래프 후보)
S400
프로그램 내부를 모듈, 파라그래프 단위로 파라그래프간의 흐름정보와 그에따른 조건들에 관한 정보 수집
S410
각 파라그래프간의 호출관계 검색
S420
호출관계 정제(Refine)
S430
비구조화 명령문에 의한 한 파라그래프 흐름 검색
S440
호출트리(Tree) 생성
S450
종료

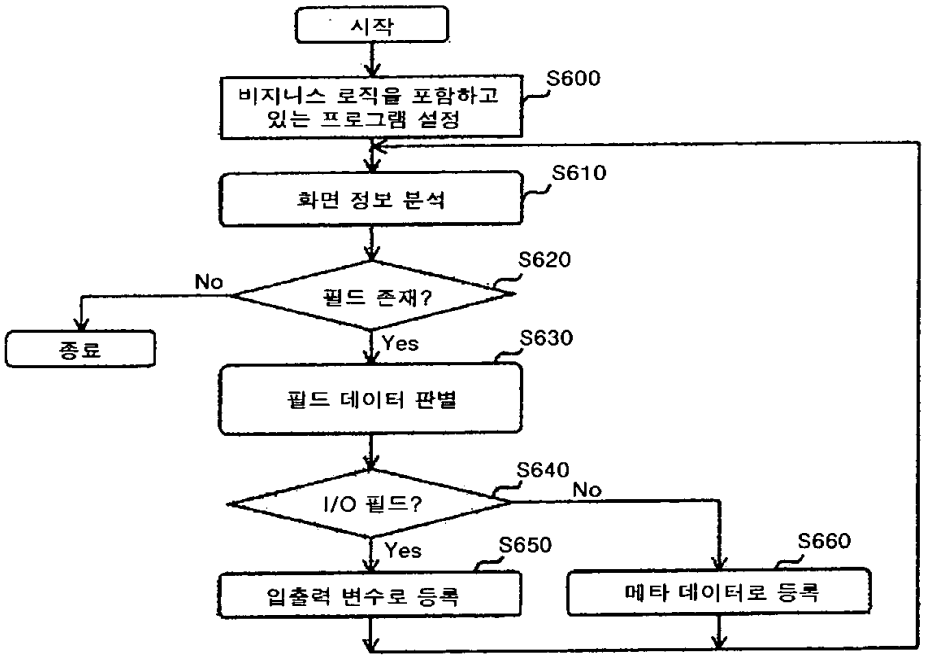
【도 5a】

```
ID                    DIVISION.
PROGRAM-ID.           KCBG0203.
ENVIRONMENT           DIVISION.
DATA                  DIVISION.
WORKING-STORAGE       SECTION.
01 REC-TERM.          COPY  KCBB0004.
   01 REC-COMMAREA.
      02    CA-CTRL-GROUP.
       03   CA-SECT-CODE      PIC 9(1).
                    ⋮

LINKAGE               SECTION.
   01 DFHCOMMAREA.
      02    DC-CTRL-GROUP     PIC X(12).
      02    DC-DATA-GROUP     PIC X(48).
                    ⋮

PROCEDURE             DIVISION.
0000-SECTION-CONTROL.
    MOVE   ZERO           TO REC-WORK-COND.
    MOVE   DFHCOMMAREA    TO REC-COMMAREA.
                    ⋮
2400-CHEORI-CONTROL.                                    }
    IF     CA-FUNC-CD-1   =  1                          }
           PERFORM        2410-INSERT-CHEORI            } 500
    ELSE                                                }
    IF     CA-FUNC-CD-1   =  2                          }
           PERFORM        2420-REPLACE-CHEORI           }
                    ⋮
2420-REPLACE-CHEORI.                                    }
    IF     CA-FUNC-CD-2   =  2                          }
           PERFORM        2421-INSERT-REPLTERM.         } 510
    MOVE   CA-BASE-KEY    TO ZT-KEY-GROUP.              }
                    ⋮
```

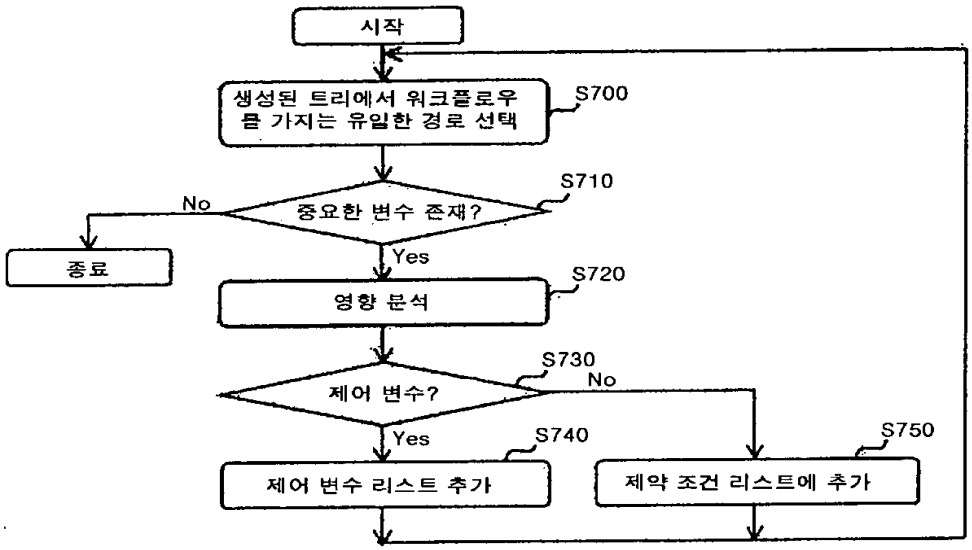
【도 5b】


S40
S50

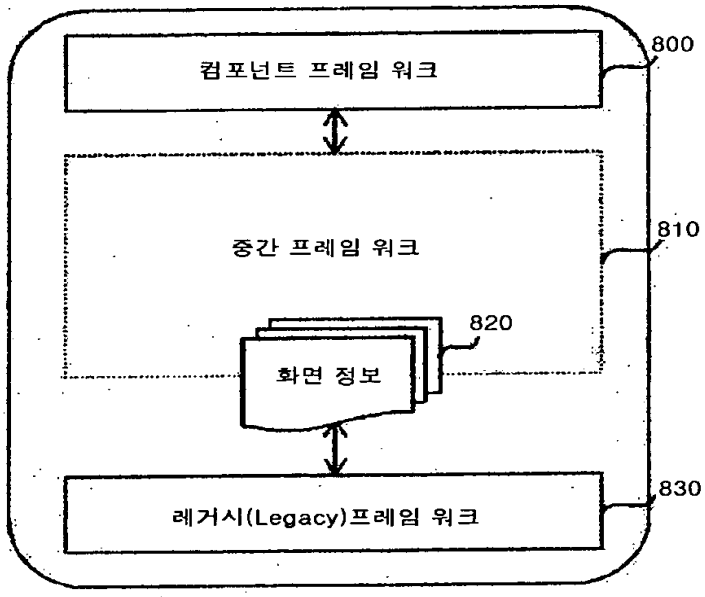
【도 6】


시작
비지니스 로직을 포함하고 있는 프로그램 설정
S600
화면 정보 분석
S610
필드 존재?
S620
No
종료
Yes
필드 데이터 판별
S630
I/O 필드?
S640
No
Yes
입출력 변수로 등록
S650
메타 데이터로 등록
S660

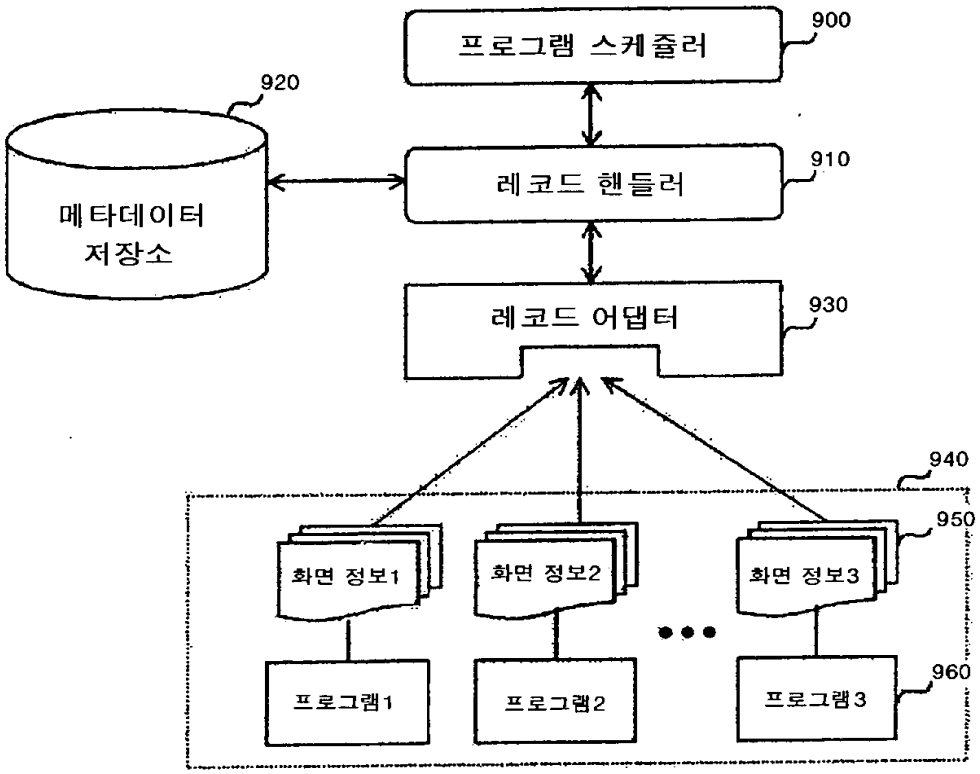
【도 7】


시작
생성된 트리에서 워크플로우를 가지는 유일한 경로 선택
S700
중요한 변수 존재?
S710
No
종료
Yes
영향 분석
S720
제어 변수?
S730
No
Yes
S740
제어 변수 리스트 추가
제약 조건 리스트에 추가
S750

【도 8】


컴포넌트 프레임 워크
800
중간 프레임 워크
810
화면 정보
820
레거시(Legacy)프레임 워크
830

【도 9】


프로그램 스케줄러
900
920
메타데이터 저장소
레코드 핸들러
910
레코드 어댑터
930
940
950
화면 정보1
화면 정보2
화면 정보3
프로그램1
프로그램2
프로그램3
960